1971

Approaches to the n-job m-machine scheduling problem.

Shackelton, Norman John.

Monterey, California ; Naval Postgraduate School

APPROACHES TO THE N-JOB M-MACHINE SCHEDULING PROBLEM

Norman John Shackelton

United States
Naval Postgraduate School

THESIS

APPROACHES TO THE N-JOB M-MACHINE SCHEDULING PROBLEM

by

Norman John Shackelton, Jr.

Thesis Advisor: Gilbert T. Howard

March 1971

Approaches to the N-Job M-Machine Scheduling Problem

by

Norman John Shackelton, Jr.
Lieutenant, United States Navy
B.S., United States Naval Academy, 1963

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1971

ABSTRACT

This thesis presents two methods of solving the n-job m-machine job shop scheduling problem. The criterion for optimality is the minimization of the total time to process all jobs on all machines. The technological ordering of machines for each job is fixed, known, and nonrandom.

The first method presented, a graphical method, indicates a lower bound and an upper bound on the optimal time to process all jobs on all machines.

The second method is a branch and bound algorithm. In principle an optimal solution can always be determined by this method. Only limited computational experience is presented for the algorithm but some methods for efficient computation are suggested.

TABLE OF CONTENTS

LIST OF DRAWINGS

I. INTRODUCTION

The problem considered is that of processing n jobs on m machines so that the time to process all jobs on all machines is a minimum. Each job is processed in turn on each machine according to a fixed machine ordering. The machine orderings for the jobs need not be the same.

The assumptions upon which solutions to the problem are based are taken from Refs. 1 and 2. They are:

1. The time to process each job on each machine is fixed, known, and nonrandom.

2. A job may not be processed by more than one machine at a time and a machine may not process more than one job at a time.

3. Once processing of a job by a machine has begun, it may not be interrupted until the processing of that job is complete.

Two approaches to the problem were considered. The problem was first represented geometrically. An attempt was then made to solve the problem graphically. The graphical method did not always provide an optimal solution but did provide an upper and lower bound on the time to complete all jobs.

The second approach to the problem was a branch and bound method of solution. In principle the branch and bound method always yields the optimal solution to the problem. Computational efficiency of the branch and bound algorithm presented has not been examined in detail, but some methods for efficient computation are suggested.

II. NATURE OF THE PROBLEM

A. DEFINITION OF A SCHEDULE

The properties of a schedule must first be defined. A machine schedule can be represented by a machine number (or letter) followed by an ordered n-tuple specifying the job ordering on that machine. Since the processing time of each job on each machine is known, m of these representations together with the known machine order for each job constitute a complete schedule. For example, if jobs 1, 2, and 3 are processed in the order 2, 3, 1 on machine A, then the representation is A:(2,3,1).

B. GRAPHICAL PRESENTATION OF A SCHEDULE

A complete schedule can be represented on a graph called a Gantt chart [Ref. 3]. A Gantt chart is simply a means of presenting a schedule visually. A list of machines is presented as the vertical edge of the chart. The horizontal edge represents time. Beside each machine is a plot of the start and finish of each job on the time axis. An example of a Gantt chart is shown in figure 1.

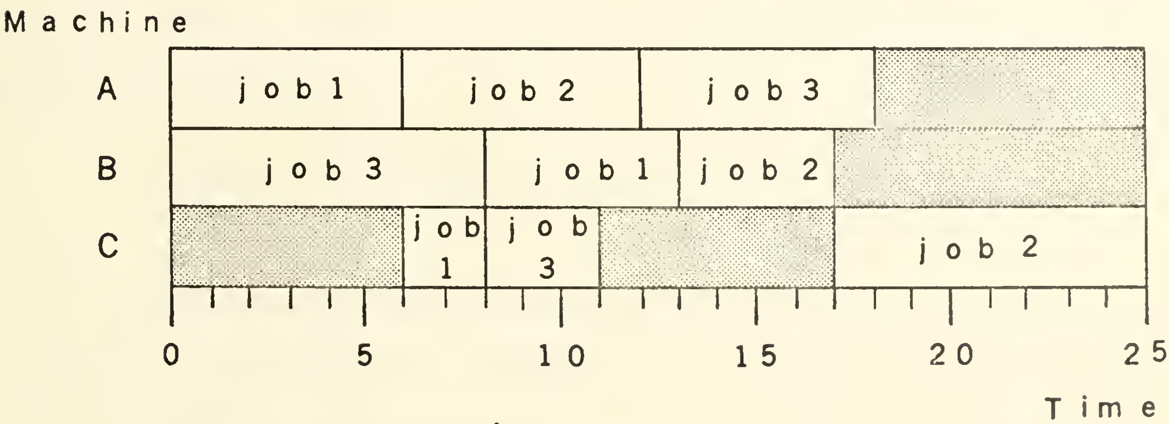


figure 1

In the example the representations A:(1,2,3), B:(3,1,2), and C:(1,3,2) are presented graphically. The machine orderings for the jobs and the time to process each job on each machine are also presented in the Gantt chart. In figure 1, job 1 must be processed first on machine A for 6 units of time, then on machine C for 2 units, and finally on machine B for 5 units of time. The machine orderings and process times for jobs 2 and 3 may be obtained in the same manner from figure 1.

C. CRITERION FOR AN OPTIMAL SCHEDULE

The criterion selected for determining an optimal schedule was to minimize the total time to process all jobs on all machines.

The time to process any job i on all machines can be represented by

$$\sum_{j=1}^{m} (t_{ij}) + s_i , \qquad i=1,\ldots,n$$

where t_{ij} = processing time for job i on machine j

s_i = total idle time of job i

m = the number of machines

and n = the number of jobs.

The problem is then to minimize the time to process all jobs on all machines or, equivalently:

$$\text{Minimize} \quad \left[\operatorname*{Max}_{i} \left\{ \sum_{j=1}^{m} (t_{ij}) + s_i, \quad i=1,\ldots,n \right\} \right] .$$

It is necessary to preserve the ordering of machines for each job, to ensure that no two jobs are processed at the same time on any machine, and to prohibit any two machines from simultaneously processing any job.

D. PREVIOUS SOLUTION METHODS

Many attempts have been made to develop a closed-form solution to the general n-job m-machine flowshop problem. Most of the solution methods devised are acceptable for small m and n but become computationally inefficient when applied to large scheduling problems.

A graphical method of solution of the 2-job m-machine problem has been proposed by Akers and Friedman [Ref. 4] and Szwarc [Ref. 5]. The graphical algorithm presented in these two papers was proved by Hardgrave and Nemhauser [Ref. 1] to provide the sequence of jobs for each machine which minimized the total time to complete both jobs.

Brooks and White [Ref. 6] and Ignall and Schrage [Ref. 7] have investigated the branch and bound method in solving the flowshop problem. Ignall and Schrage's algorithm was applied to the case where the criterion for optimality was the minimization of the time to complete all jobs but the machine ordering for each job was identical. Brooks and White have stated that their procedure uses a great deal of computation time on a computer. No detailed computational experience was presented for their problem.

Integer programming has also been attempted as a method of solving the jobshop problem but, according to Conway, Maxwell, and Miller [Ref.3], very little progress has been made. The reasons for lack of success in this area appear to be the large size of the integer programs involved and the large time required to solve the resultant integer programming problem.

III. GRAPHICAL SOLUTION TECHNIQUES

A. HARDGRAVE-NEMHAUSER GRAPHICAL ALGORITHM

1. Two-Job m-Machine Scheduling Problem

Although Akers and Friedman [Ref. 4] and Szwarc [Ref. 5] have described the graphical method of solution, Hardgrave and Nemhauser [Ref. 1] have provided a rigorous proof that the algorithm leads to an optimal solution. For this reason, the graphical method is referred to as the Hardgrave-Nemhauser graphical algorithm.

a. The Geometric Model

A geometric model of the 2-job m-machine problem is used in the Hardgrave-Nemhauser graphical algorithm. A coordinate axis is drawn with the ordinate representing the time to complete one job on all machines, the abscissa the time to complete the other. The known machine orderings and their associated processing times are marked along the axis corresponding to the appropriate job. Lines are drawn perpendicular to the axes at the finish times of the machine processing operations. The rectangles formed by the intersection of the spaces corresponding to a particular machine for each of the two jobs are shaded in (as in figure 2).

From any point a line can be drawn parallel to or at a 45 degree angle with the axes. A line of this type from (0,0) to (S_1,S_2) that does not pass through any infeasible region (shaded area) is called a feasible path. A feasible path specifies an ordering of jobs on machines. That is, if the path passes beneath an infeasible region, then the job on the ordinate is processed before the job on the abscissa on

ORDER OF OPERATIONS (PROCESSING TIME)

JOB 1: A(2), B(4), C(3), D(1)

JOB 2: D(3), A(2), C(3), B(2)

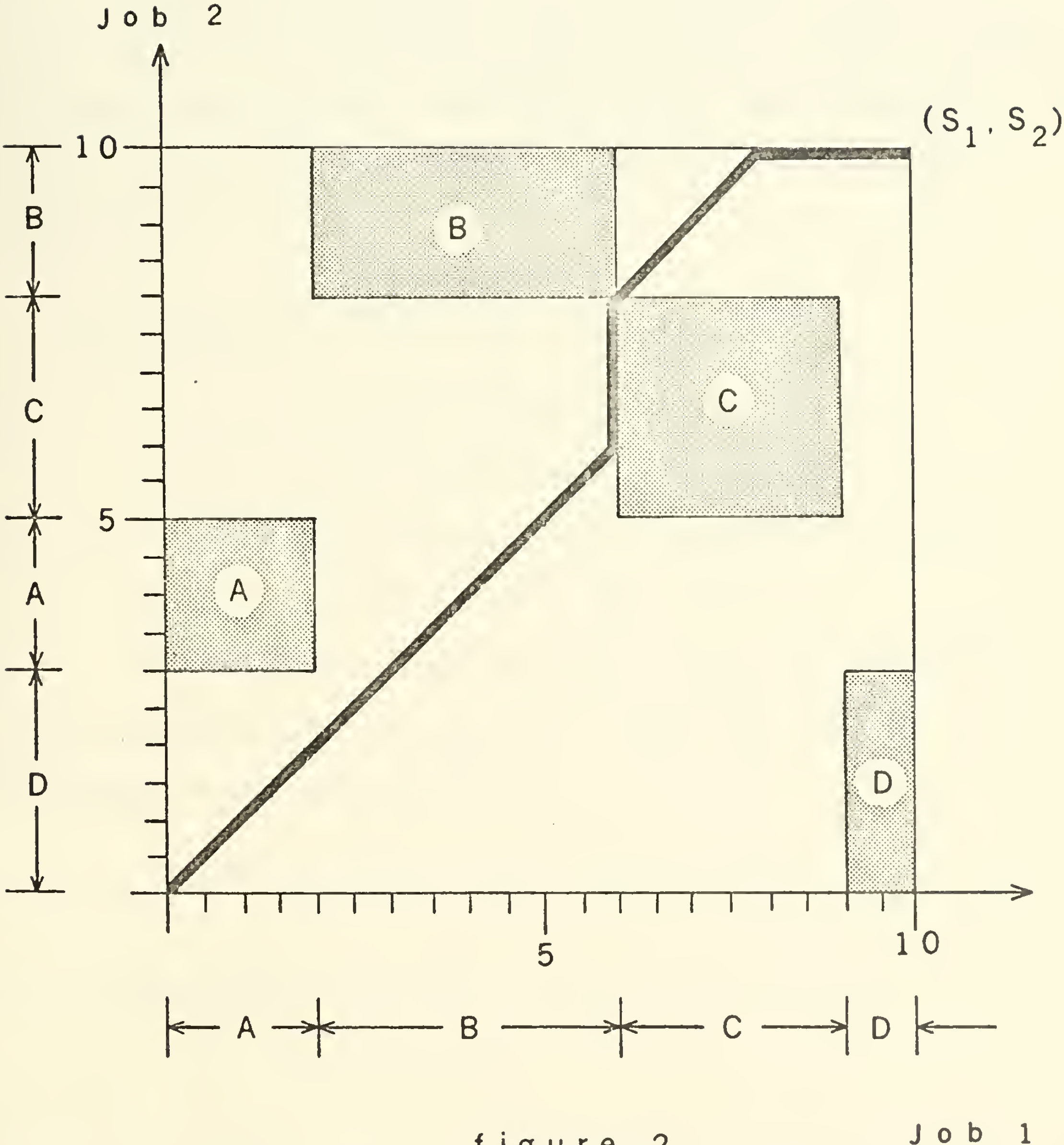


figure 2

the machine generating that infeasible region. Then, a feasible path from (0,0) to (S_1,S_2) specifies a complete schedule.

Hardgrave and Nemhauser proved that the shortest feasible path from (0,0) to (S_1,S_2) yields the schedule which minimizes total process time of all jobs on all machines. Note that travel for one unit of time in any direction indicates the manner in which jobs are processed. Travel in a vertical direction indicates that only job 2 is being processed, in a horizontal direction that only job 1 is being processed, and diagonal travel indicates simultaneous processing of both jobs. The length of the shortest feasible path, λ_1, from (0,0) to (S_1,S_2) was proved to be:

$$\lambda_1 = s_{1m} + p_{1m} + \text{Max}(S_1 - s_{1m} - p_{1m}, S_2 - s_{2m})$$

where s_{im} = set up time on machine m for job i, i=1,2

p_{im} = processing time on machine m for job i, i=1,2

and m is the machine generating the infeasible region that causes a conflict between jobs 1 and 2.

The equation for shortest path is based on the assumption that job 1 precedes job 2 on machine m. If job 2 precedes job 1, then the equation for the shortest path is

$$\lambda_1 = s_{2m} + p_{2m} + \text{Max}(S_2 - s_{2m} - p_{2m}, S_2 - s_{2m})$$

and the optimal ordering of jobs on machine m can be determined from

$$\lambda^* = \text{Min}(\lambda_1,\lambda_2).$$

That is, if $\lambda_1 < \lambda_2$ then job 1 precedes job 2 on machine m. If

$\lambda_2 < \lambda_1$ then job 2 precedes job 1 on m. Finally if $\lambda_1 = \lambda_2$ then either can occur and there is an alternate optimal solution to the problem.

b. The Graphical Algorithm

The graphical algorithm of Hardgrave and Nemhauser consists of rules for branching the path around infeasible regions. The algorithm is:

STEP 1: Starting at (0,0) move diagonally until an infeasible region is encountered, say on its bottom edge at point (t, s_{2m}). (If the left side of the infeasible region is encountered, the rules are similar.)

STEP 2: Branch in two directions around the infeasible region: (a) Horizontally along the bottom edge until the point $(s_{1m} + p_{1m}, s_{2m})$ is reached. The return to STEP 1, using $(s_{1m} + p_{1m}, s_{2m})$ as the new starting point. (b) From point (t, s_{2m}) move in reverse along the path to point $(s_{1m}, s_{2m} - t + s_{1m})$. From this point move vertically along the left edge of the infeasible region until point $(s_{1m}, s_{2m} + p_{2m})$ is reached. The return to STEP 1 using this as a starting point.

STEP 3: When the top or right edge of the outer rectangle is encountered, move along that edge to the finish point (S_1, S_2). The shortest path is the smallest of those paths generated in this manner. From this path the optimal schedule is found.

2. The n-Job m-Machine Problem

The geometric model and the graphical algorithm were extended to the general case by Hardgrave and Nemhauser. The model became a graph with n axes, one for each job. The infeasible regions in the model were n-dimensional polyhedrons. An optimal solution was characterized by the

shortest line which passed from $(0,\ldots,0)$ to $(S_1,\ldots,S_n)$ without passing through any of the infeasible regions. Branching rules were given in the paper. Even for $n=3$, this procedure was difficult to actually perform.

B. GRAPHICAL BOUNDS ON OPTIMAL PATH LENGTH

1. Description of the Method

The $\binom{n}{2}$ 2-dimensional orthogonal projections of the n-dimensional polyhedrons are simply rectangles. The projections of the entire n-dimensional graph are $\binom{n}{2}$ 2-dimensional graphs. If these 2-dimensional graphs are looked upon as the graphs associated with $\binom{n}{2}$ 2-job m-machine problems, the shortest path in each graph may be found in the manner described previously. Each of these shortest paths indicates the ordering of two jobs on each machine. Considering all of the 2-job solutions, there are $\binom{n}{2}$ pairwise orderings of all the jobs on all the machines. These pairwise orderings of jobs on machines indicate an ordering of all jobs on all machines. This ordering of jobs on each machine may be the plotted as a schedule on a Gantt chart as in figure 1, since the other required information is known (i.e., job's machine ordering and processing times). The time to complete the schedule is then determined by completion of the Gantt chart.

W. Szwarc [Ref. 2] has also arrived at this technique and called it an approximate solution to the n-job m-machine scheduling problem. He showed that, under certain conditions, this method yielded an optimal solution to the problem in which each of the jobs had the same machine orderings.

The method described does not always guarantee a feasible ordering of jobs on machines. Likewise, if the ordering of all jobs on all

machines obtained is feasible, there is no guarantee that the ordering is optimal. In other words, the time to completion indicated by the Gantt chart is not necessarily the minimum time to complete all jobs on all machines. Of course the solution obtained in this way is sometimes optimal.

2. Feasibility

The ordering of jobs on a machine A is infeasible if, for any of the three jobs i, j, and k, the pairwise orderings take the form A:(i,j), A:(j,k), A:(k,i).

If the ordering of jobs on some machine is not feasible, the ordering may be made feasible by applying the following branching rule:

For one machine suppose the ordering of jobs is infeasible. That is, for some machine A the pairwise orderings for three jobs i, j, and k are of the form A:(i,j), A:(j,k), A:(k,i). Then, in each of the three associated graphs, branch the path in each of the 2^n possible ways around the rectangle causing the infeasibility. That is, generate the orderings

(1) A:(i,j), A:(j,k), A:(i,k)

(2) A:(i,j), A:(j,k), A:(k,i)

(3) A:(i,j), A:(k,j), A:(i,k)

(4) A:(i,j), A:(k,j), A:(k,i)

(5) A:(j,i), A:(j,k), A:(i,k)

(6) A:(j,i), A:(j,k), A:(k,i)

(7) A:(j,i), A:(k,j), A:(i,k)

and (8) A:(j,i), A:(k,j), A:(k,i).

Of these orderings, (2) is the original infeasible ordering and (7) is also an infeasible ordering. Therefore, these orderings are not

considered. First, choose those orderings that do not cause any of the other orderings to become infeasible. Next, among those orderings, choose the one which causes the least increase in each of the shortest paths in the three associated graphs. Although nothing can be said regarding the optimality of this generated path, it is feasible. Certainly if n is large this method is not practical.

3. Upper and Lower Bounds

The time to complete the feasible schedule generated by the solution of the $\binom{n}{2}$ 2-job m-machine problems clearly provides an upper bound on the minimum time to process all jobs on all machines. Call this time λ_u. A lower bound can also be found, thereby providing an upper and a lower bound on the optimal solution.

Let λ_{ij} = the shortest path for the 2-job m-machine problem with coordinate axis labeled i and j.

and λ^* = the minimum time to process all jobs on all machines.

Then, $$\lambda^* \geq \lambda_{ij}, \text{ for each } i,j,\ i \neq j$$

or $$\lambda^* \geq \max_{i \neq j} (\lambda_{ij}),\ i,j = 1,\ldots,n.$$

Then, $\lambda_l = \max_{i \neq j} (\lambda_{ij})$ is a lower bound on λ^*, or

$$\lambda_l \leq \lambda^* \leq \lambda_u \ .$$

Clearly, if $\lambda_u = \lambda_l$ then $\lambda^* = \lambda_u = \lambda_l$ and the optimal schedule has been generated. In most cases, however, this will not be the case and the method merely indicates an upper and a lower bound on the solution.

Whether the upper bound is a tight bound or not, it can be used to eliminate searching some of the branches in the branch and bound method of section IV.

4. An Example

As an illustration of the method, consider the 3-job 4-machine problem with the machine orderings:

	Job	Machine (process time)
	1	A(3), B(3), C(7), D(6)
	2	B(5), A(6), D(2), C(2)
and	3	A(4), C(2), D(3), B(4).

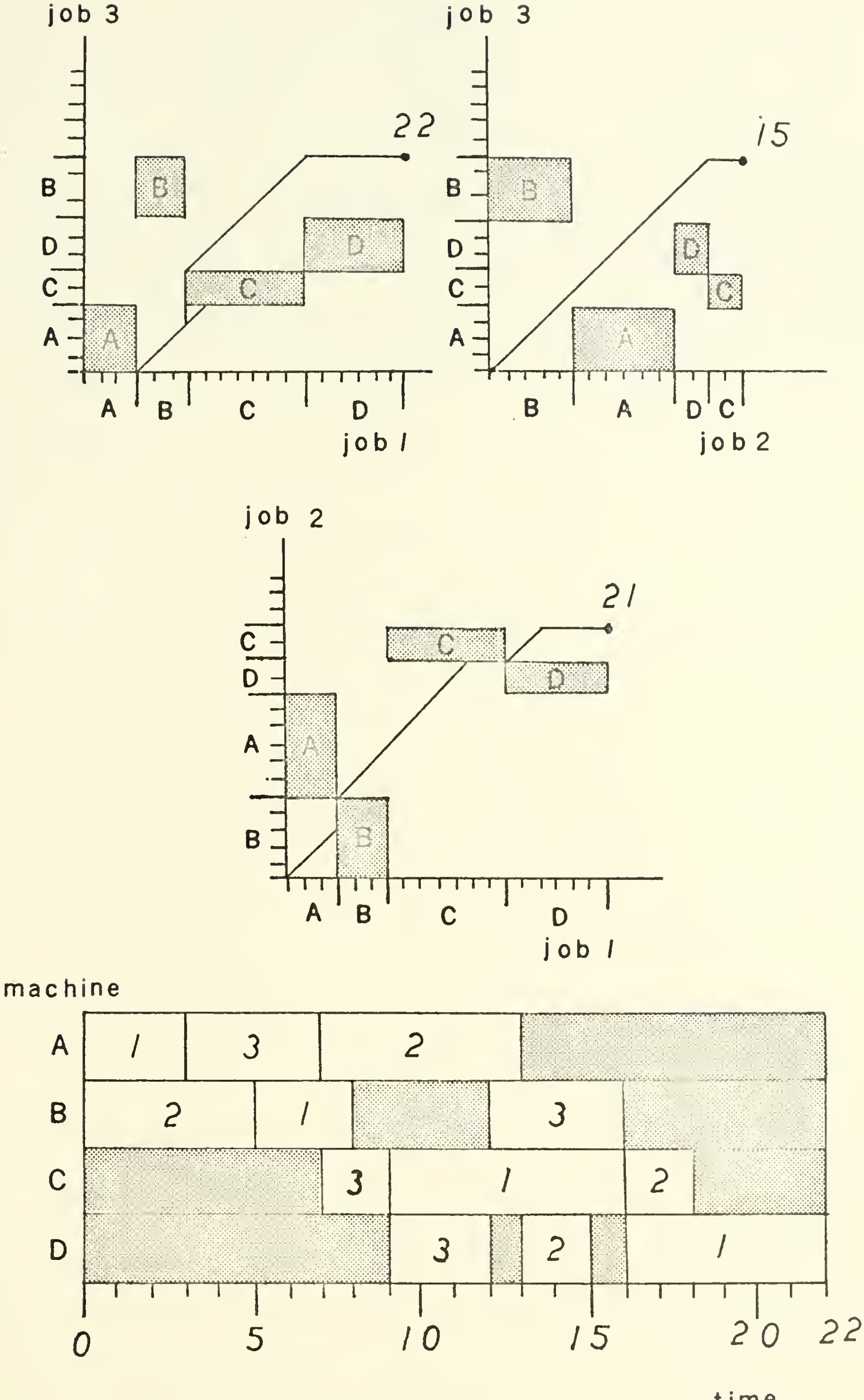
The problem is to minimize the time to process jobs 1, 2, and 3 on machines A, B, C, and D. Since n=3, there are $\binom{3}{2} = \frac{3!}{1!2!} = 3$ 2-job 4-machine problems to be solved graphically. The solutions to the three problems are shown in figure 3. The job orderings obtained from these three graphical problems are:

(1) Job 1 vs. Job 3, $\lambda_{13}=22$, A:(1,3)
B:(1,3)
C:(3,1)
D:(3,1)

(2) Job 2 vs. Job 3, $\lambda_{23}=15$, A:(3,2)
B:(2,3)
C:(3,2)
D:(3,2)

(3) Job 1 vs. Job 2, $\lambda_{12}=21$, A:(1,2)
B:(2,1)
C:(1,2)
D:(2,1).

The aggregate job orderings formed from these three pairwise orderings are:



job 3
22
B
D
C
A
A
B
C
D
job 1
job 3
15
B
D
C
A
B
A
D
C
job 2
job 2
21
C
D
A
B
A
B
C
D
job 1
machine
A
1
3
2
B
2
1
3
C
3
1
2
D
3
2
1
0
5
10
15
20
22
time

figure 3

$$A:(1,3,2)$$

$$B:(2,1,3)$$

$$C:(3,1,2)$$

and $$D:(3,2,1).$$

These orderings are all feasible. The Gantt chart is drawn and the upper bound on the time to process all jobs is obtained (λ_u=22). The lower bound is λ_l=Max(λ_{13}, λ_{23}, λ_{12})=22. Since $\lambda_l=\lambda_u$=22, the solution λ^*=22 is optimal and the optimal schedule is presented in the Gantt chart in figure 3.

IV. BRANCH AND BOUND SOLUTION

A. PROBLEM FORMULATION

Consider again the problem of processing n jobs on m machines so that the time to process all jobs on all machines is a minimum. The machine ordering for each job is fixed and is known, as is also the processing time for each job on each machine.

As stated previously, the total time to process any job i on all machines can be represented by

$$\sum_{j=1}^{m} (t_{ij}) + s_i, \quad i = 1,\ldots,n$$

where t_{ij} = the processing time for job i on machine j,

s_i = total idle time of job i,

m = the number of machines,

and n = the number of jobs.

The problem is then to minimize the time to process all jobs on all machines. This can be written as

$$\text{Minimize } \left[\underset{i}{\text{Max}} \left\{ \sum_{j=1}^{m} (t_{ij}) + s_i, \; i = 1,\ldots,n \right\} \right] .$$

At the same time it is required that the ordering of machines for each job be preserved, no two jobs be processed at the same time on any machine, and no job be simultaneously processed by any two machines.

The total processing time for job i can also be represented as

$$\sum_{j=1}^{m-1} (t_{ij}) + s_i + t_{im}$$

where m is the number of the last machine, in order, to process job i and the other variables are as defined previously. Then, setting

$$S_{im} = \sum_{j=1}^{m-1} (t_{ij}) + s_i ,$$

a new form of the objective function is

$$\text{Minimize} \quad [\ \underset{i}{\text{Max}} \ \{ S_{im} + t_{im}, \ i = 1,\ldots,n \ \} \ ] .$$

S_{im} is called the earliest start time of job i on machine m, or, the earliest start time of the last process which job i must undergo.

Let F_{i1} be the latest completion time of job i on machine 1. Since job i and job j cannot be processed at the same time on any machine 1, then

$$S_{i1} \geq F_{j1}$$

if job j precedes job i on machine 1.

Additionally, if job i is to be processed on machine 1 before being processed on machine k, then

$$S_{ik} \geq F_{i1}$$

The original problem can now be restated as

$$\text{Minimize} \quad [\ \underset{i}{\text{Max}} \ \{ S_{i,i_m} + t_{i,i_m}, \ i = 1,\ldots,n \ \} \ ] .$$

Subject to $\quad S_{i1} \geq F_{j1}$, if j precedes i on 1

or $\quad S_{j1} \geq F_{i1}$, if i precedes j on 1, for all i, j=1,...,n

$1=i_1,\ldots,i_m$

and

$$S_{i,i_2} \geq F_{i,i_1}$$

$$S_{i,i_3} \geq F_{i,i_2}$$

$$\vdots$$

$$S_{i,i_m} \geq F_{i,i_{m-1}} \quad , \text{ for all } i$$

where the machine ordering specified for job i is $i_1,\ldots,i_m$.

B. BRANCH AND BOUND ALGORITHM

1. Branching Method

The problem, as stated, can be solved using a branch and bound algorithm similar to that used by Little and others [Ref. 8] to solve the traveling salesman problem. In the job shop scheduling problem a branching operation can be defined through the ordering of jobs on machines. First consider job 1 to be scheduled on all machines. Then define a branch by ordering job 2 and job 1 on the first machine to process job 2. These orderings would be $2 < 1$ and $1 < 2$. Then calculate a lower bound on the time to process all jobs on all machines (i.e., $\text{Max}\ (S_{i,i_m} + t_{i,i_m},\ i=1,\ldots,n)$). From the node giving the smallest lower bound, branch again in the same manner for the next machine to process job 2. Continue branching in this way until both jobs 1 and 2 are scheduled on all machines. When this has been done, branch in three ways from the node with the smallest lower bound. That is, branch on the orderings $3 < 1 < 2$, $1 < 3 < 2$, and $1 < 2 < 3$ for the first machine to process job 3. These are the orderings if job 1 preceded job 2 on this machine. If on the contrary, job 2 preceded job 1

on the machine in question, then the orderings would be $3 < 2 < 1$, $2 < 3 < 1$, and $2 < 1 < 3$. Then calculate the lower bound and continue to branch on the smallest existing lower bound. Continue to add jobs until all jobs have been processed on all machines. When the smallest of the lower bounds has been found and all jobs have been processed on all machines, the optimal solution has been found.

An illustration of the branching method is shown in figure 4. In the illustration, the set of all job-machine orderings is partitioned by ordering job 1 and job 2 on machine A. This is represented by $2 < 1$ and $1 < 2$. The fixed ordering of machines specified for job number 2 is used for the two job partitioning. In the illustration, suppose that the ordering of machines A, B, C is the ordering specified for job 2. Job 1 must also be processed according to its specified ordering on machines. This is done by considering job 1 to be scheduled already. The ordering of jobs may be the same or different. By branching on successive orderings of jobs on machines, this technique is capable of representing any finite number of jobs and machines. It is limited by the number of computations.

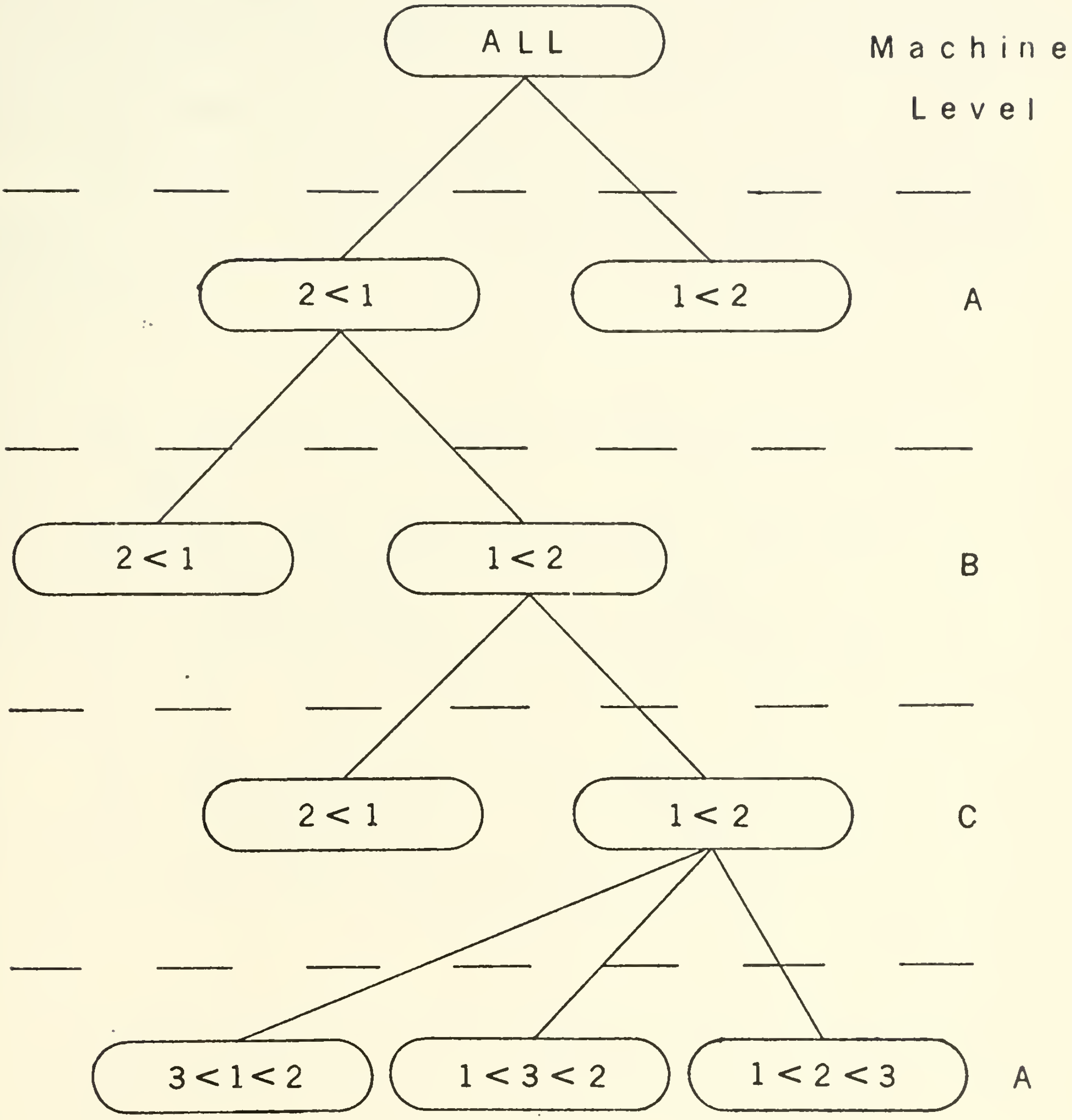
2. Lower Bound Computation and Algorithm

If a job j follows a job i on machine 1, then it must be true that

$$S_{j1} \geq F_{i1}.$$

Likewise, if job i is processed first on machine k and then on machine 1, then

$$S_{i1} \geq F_{ik}.$$



BRANCHING METHOD

ORDERING OF JOBS ON MACHINES

figure 4

It is possible to record these times in tableau form and, consequently, to know the time required to process all jobs on all machines (through the present branch). This time is $S_{i,i_m} + t_{i,i_m}$, where i_m is the last machine encountered by job i.

The tableau to be used is one of the form illustrated in figure 5, where $T_i = S_{i,i_m} + t_{i,i_m}$, $i = 1,\ldots,n$ and S_{ij} and F_{ij} are as previously defined.

job \ machine	1	2	. . .	m	T_i
1	S_{11} / F_{11}	S_{12} / F_{12}	. . .	S_{1m} / F_{1m}	T_1
⋮	⋮	⋮	. . .	⋮	⋮
n	S_{n1} / F_{n1}	S_{n2} / F_{n2}	. . .	S_{nm} / F_{nm}	T_n

figure 5

The methods of computing the required lower bound and describing the branching operations is given in the following algorithm.

STEP 1: For the initial tableau, fill in the blocks of the tableau as if each job were to be processed independently of every other job. When completed, every row i should have entries of the form

$$F_{ik} = S_{i,k+1}, \quad i = 1,\ldots,n, \quad k = i_1,\ldots,i_{m-1}$$

and

$$F_{i,i_m} = T_m, \quad i = 1,\ldots,n.$$

This schedule will undoubtedly be infeasible for joint processing of all the jobs for any interesting problem. If this schedule is feasible, then it is optimal and the minimum time to process all jobs is

$$\underset{i}{\text{Max}} \{ T_i, \ i = 1,\ldots,n \} .$$

If this schedule is infeasible, go to STEP 2.

STEP 2: For the first partition (suppose it is $i < j$, or i precedes j on machine k) evaluate the lower bound on

$$\underset{i}{\text{Max}} \{ T_i, \ i = 1,\ldots,n \} .$$

It is known from the constraints that for job i to precede job j on machine k, it is necessary that

$$S_{jk} \geq F_{ik} .$$

The procedure, therefore, is:

(a) If $F_{ik} > S_{jk}$, add $F_{ik} - S_{jk}$ to S_{jk} and to every S_{jl} where l follows k in the machine ordering for job j. That is, enter in the next tableau:

$$\hat{S}_{jl} = F_{ik} - S_{jk} + S_{jl}, \text{ for all l where } S_{jl} \geq S_{jk}$$

$$\hat{T}_j = \hat{S}_{j,j_m} + t_{j,j_m}$$

and $$\hat{F}_{jl} = \hat{S}_{jl} + t_{jl}, \text{ for all l where } S_{jl} \geq S_{jk}.$$

The lower bound is

$$\underset{i}{\text{Max}} \{ T_i, \ i = 1,\ldots,n \} .$$

(b) If $F_{ik} \leq S_{jk}$, make no change. Record the lower bound, $\underset{i}{\text{Max}} \{T_i\}$.

STEP 3: When the lower bound for the branch $i < j$ has been found, compute the lower bound for the branch $j < i$ in the same manner.

STEP 4: Compare the lower bounds thus obtained and continue to branch along the path with the lowest computed lower bound. At each iteration, compare the latest finish time of the first job in the ordering with the earliest start time of the second job in the ordering. Then, if any change is necessary, compute the new $\hat{S}_{jl}$, $\hat{T}_j$, and $\hat{F}_{jl}$ and compare that $\hat{F}_{jl}$ with the next job's S_{kl} in the precedence ordering for that machine. Continue this procedure until, for the machine level reached, all

$$S_{il} \geq F_{jl}, \text{ if } j<i \text{ on } l$$

$$\text{or } S_{jl} \geq F_{il}, \text{ if } i<j \text{ on } l$$

$$\text{for all } i, j = 1,\ldots,n$$

$$\text{and } l = i_1,\ldots,i_m$$

and

$$S_{i,i_2} \geq F_{i,i_1}$$

$$\vdots$$

$$S_{i,i_m} \geq F_{i,i_{m-1}}, \text{ for all } i = 1,\ldots,n.$$

Record the lower bound, $\max_i \{T_i\}$.

STEP 5: Continue to branch along the paths for which the least lower bound has been recorded until all jobs have been ordered on all machines. Then if there is no recorded lower bound in any branch that is less than that finally computed, the optimal solution is specified.

3. An Example

Consider again the example problem of scheduling three jobs, 1, 2,

and 3, on four machines, A, B, C, and D. The machine orderings and processing times are:

Job	Machine (process time)
1	A(3), B(3), C(7), D(6)
2	B(5), A(6), D(2), C(2)
3	A(4), C(2), D(3), B(4).

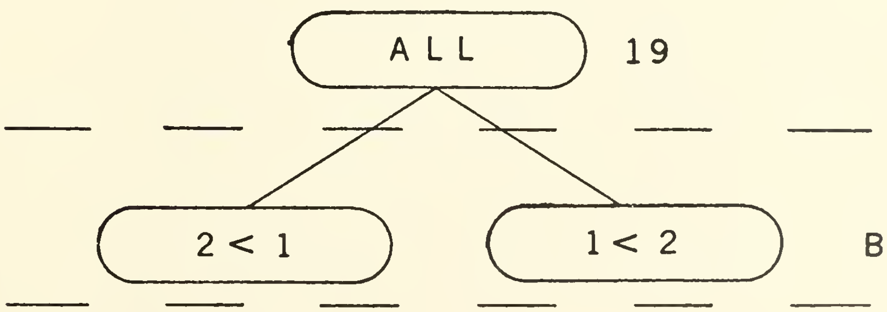
The first tableau is

(1)

	A	B	C	D	T_i
1	0 / 3	3 / 6	6 / 13	13 / 19	19
2	5 / 11	0 / 5	13 / 15	11 / 13	15
3	0 / 4	9 / 13	4 / 6	6 / 9	13

$\underset{i}{\mathrm{Max}} \{ T_i \}$

On the first branch, since job 2's machine ordering is B, A, D, C, the set of all orderings is partitioned into $2 < 1$ and $1 < 2$ on machine B. Illustrated, this is



Suppose the branch $2 < 1$ is checked first. As described in the algorithm, F_{2B} is compared with S_{1B}. Note that $F_{2B} = 3$. That is, $F_{2B} > S_{1B}$. Therefore a change in the tableau is made of the form:

$$\hat{S}_{jl} = F_{ik} - S_{jk} + S_{jl}, \text{ for all } l \text{ where } S_{jl} \geq S_{jk}$$

$$\hat{T}_j = \hat{S}_{j,j_m} + t_{j,j_m}$$

and $$\hat{F}_{jl} = \hat{S}_{jl} + t_{jl}, \text{ for all } l \text{ where } S_{jl} \geq S_{jk}.$$

Or, since B is the second machine in job 1's ordering, $l = B,C,D$.

Therefore

$$\hat{S}_{1B} = F_{2B} - S_{1B} + S_{1B} = 5$$
$$\hat{S}_{1C} = F_{2B} - S_{1B} + S_{1C} = 8$$
$$\hat{S}_{1D} = F_{2B} - S_{1B} + S_{1D} = 15$$
$$\hat{T}_1 = \hat{S}_{1D} + t_{1D} = 21$$
$$\hat{F}_{1B} = \hat{S}_{1B} + t_{1B} = 8$$
$$\hat{F}_{1C} = \hat{S}_{1C} + t_{1C} = 15$$

and $$\hat{F}_{1D} = \hat{S}_{1D} + t_{1D} = 21$$

The new tableau is then

(2)

	A	B	C	D	T_i	
1	0 / 3	5 / 8	8 / 15	15 / 21	21	$\max_i \{T_i\}$
2	5 / 11	0 / 5	13 / 15	11 / 13	15	
3	0 / 4	9 / 13	4 / 6	6 / 9	13	

At this iteration $\max_i \{T_i\} = 21$. Record this as the lower bound and check the branch $1 < 2$. The same procedure is followed and the tableau is

(3)

	A	B	C	D	T_i
1	0 / 3	3 / 6	6 / 13	13 / 19	19
2	11 / 17	6 / 11	19 / 21	17 / 19	21
3	0 / 4	9 / 13	4 / 6	6 / 9	13

$\max_i \{T_i\}$

Since both computed lower bounds are the same, the next branch can be made from either. Suppose $1 < 2$ on B is chosen and partitioned again into $2 < 1$ and $1 < 2$ on A. Note that job 2 cannot precede job 1 on machine A with the specification that job 1 precede job 2 on machine B. That branch can, therefore, be omitted. A lower bound of ∞ is then assigned. In the example, branch $1 < 2$ on A is then made. Noting that $F_{1A} \leq S_{2A}$, no change is made in the tableau, but another branch is made, partitioning on $2 < 1$ and $1 < 2$ on D. This time, for $2 < 1$, $F_{2D} > S_{1D}$ and a tableau change is made. The new $\max_i \{T_i\} = 25$. Likewise, for the branch $1 < 2$ on D, it is found that $\max_i \{T_i\} = 23$. Since both of these bounds exceed 21 (the bound on $2 < 1$ on A), that bound determines the next branch. It was not necessary to compute other than $\hat{T}_2 = 25$ or $\hat{T}_1 = 23$. If all the bounds computed exceed $\hat{T}_1$, the new tableau could then be reconstructed.

Branching on $2 < 1$ on A, it is noted that $\hat{T}_1 = 32$, so the branch $1 < 2$ on A can be checked. It is noted that $F_{1A} < S_{1A}$ so the same tableau is applicable for the next branch and the lower bound is still 21. The computations are continued in this manner until the two jobs are scheduled on all machines. Then machine 3 must be considered. The computational method is the same except that there are now three jobs to be considered. The branching and bounding is continued until the branch $2 < 1 < 3$ on B is reached and its lower bound is computed. Since this lower bound is 22, a number smaller than all other computed lower bounds, an optimal solution has been found. The final tableaux are

(4)

	A	B	C	D	T_i
1	0 / 3	5 / 8	8 / 15	15 / 21	21
2	5 / 11	0 / 5	15 / 17	11 / 13	17
3	0 / 4	9 / 13	4 / 6	6 / 9	13

$\max_i \{ T_i \}$

(5)

	A	B	C	D	T_i
1	0 / 3	5 / 8	8 / 15	15 / 21	21
2	7 / 13	0 / 5	15 / 19	13 / 15	19
3	3 / 7	12 / 16	7 / 9	9 / 12	16

$\max_i \{ T_i \}$

(6)

	A	B	C	D	T_i
1	0 / 3	5 / 8	9 / 16	16 / 22	22
2	7 / 13	0 / 5	16 / 18	13 / 15	18
3	3 / 7	12 / 16	7 / 9	9 / 12	16

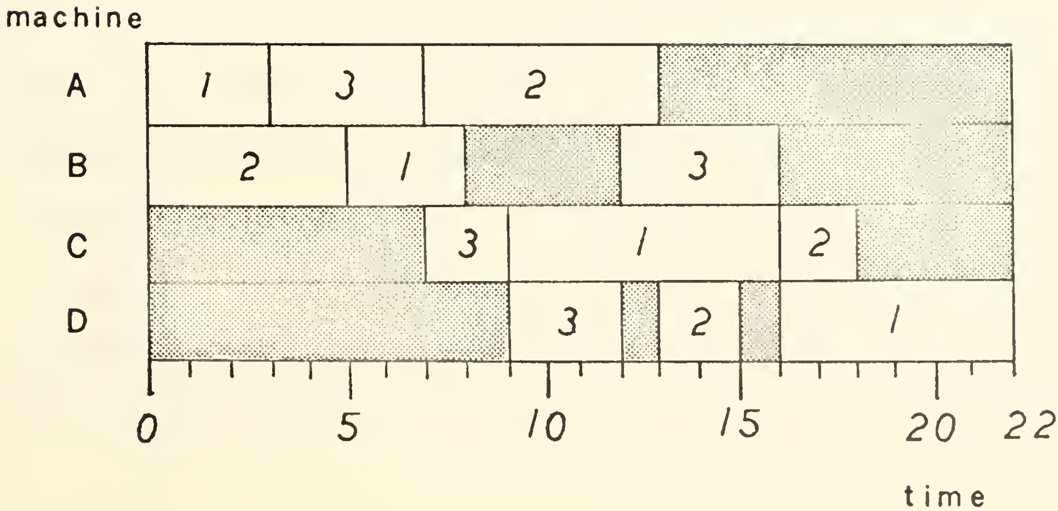
$\underset{i}{\text{Max}} \{T_i\}$

(7)

	A	B	C	D	T_i
1	0 / 3	5 / 8	9 / 16	16 / 22	22
2	7 / 13	0 / 5	16 / 18	13 / 15	18
3	3 / 7	12 / 16	7 / 9	9 / 12	16

$\underset{i}{\text{Max}} \{T_i\}$

When presented in the form of a Gantt chart, this schedule is



The complete solution tree is illustrated in figure 6. The numbers in parentheses beside the tableaux illustrate the correspondence between the branches in figure 6 and the tableaux.

C. SUGGESTIONS FOR EFFICIENT COMPUTATION

The branch and bound algorithm presented clearly yields the optimal solution to the n-job m-machine scheduling problem. Its computational efficiency, however, has not been examined. An accurate statement regarding its usefulness in solving a real problem, therefore, cannot presently be made. Some possible refinements to the algorithm are suggested.

At each lower bound computation the only arithmetic operations performed are addition and subtraction. These operations take place only when required by the comparisons of job start and finish times. The number of additions can possibly be reduced by computing only as many $\hat{T}_j$'s as necessary. That is, when any $\hat{T}_j$ is greater than a lower bound that has been computed, cease calculation on that branch and proceed to a branch which appears to offer the best possibilities. In other words, proceed to the branch with the smallest lower bound presently computed. It may be necessary to return to one of these partially completed branches. Storage availability and computation expense would determine the tradeoff between storing a partially completed tableau and performing all the arithmetic necessary to reproduce a tableau.

Another possible refinement of the method involves the use of the graphically computed upper bound of section III. B. This upper bound can be used to eliminate entire branches from the branch and bound

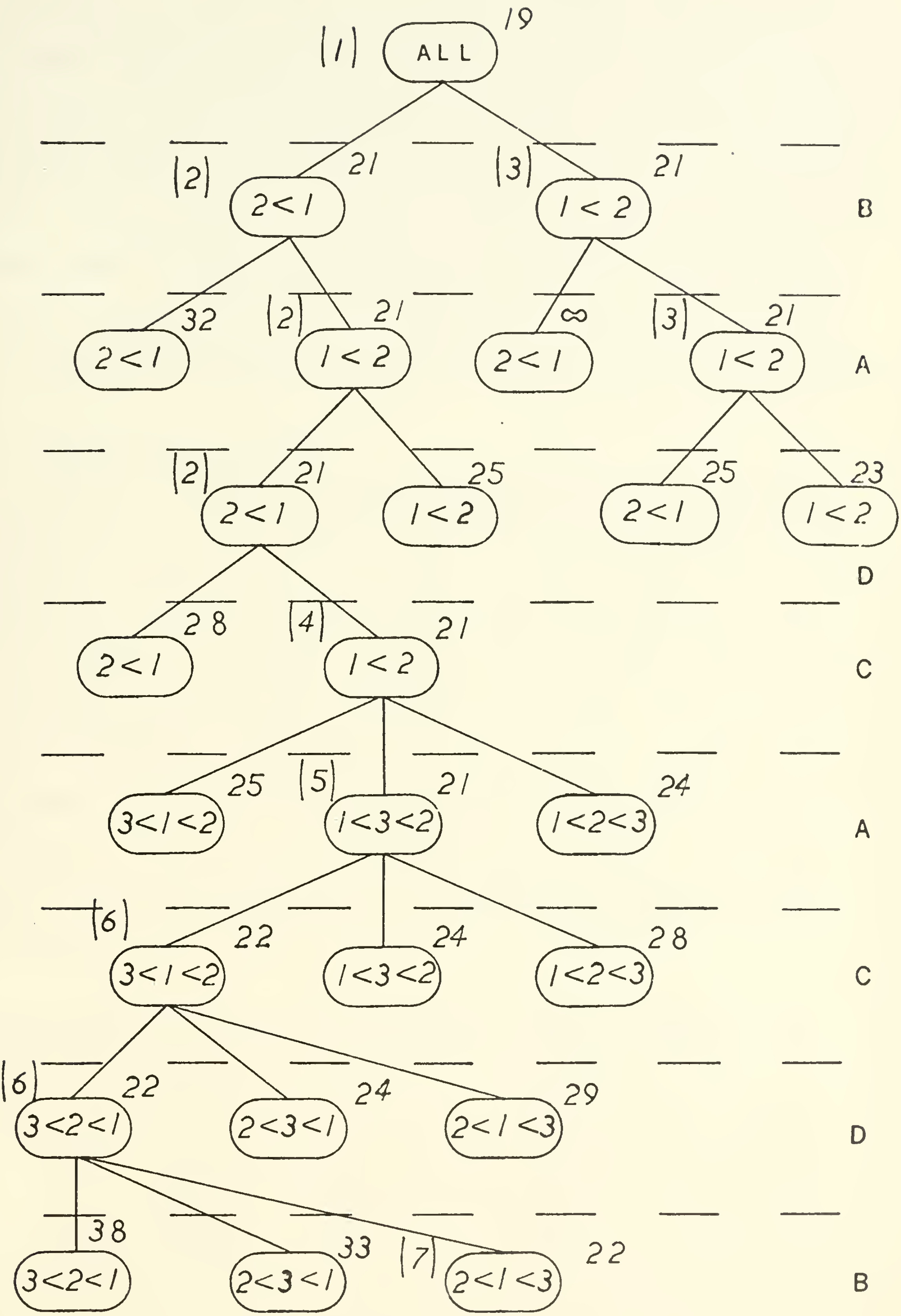

ALL
19
(1)
2<1
1<2
21
(2)
(3)
B
32
∞
25
23
28
(4)
A
D
C
3<1<2
1<3<2
1<2<3
(5)
24
22
(6)
3<2<1
2<3<1
2<1<3
29
38
33
(7)

figure 6

solution tree. As the upper bound computed in this manner represents the time to complete one feasible schedule, any lower bound discovered to be greater than this time may be disregarded. The branch and bound procedure clearly will not permit return to a branch with a lower bound greater than the optimal schedule. The optimal schedule will not be known, however, until the algorithm has terminated. The use of the graphical upper bound can then eliminate the requirement for storing any information connected with the branches with excessive lower bounds.

D. COMPUTATIONAL EXPERIENCE

A computer program for the branch and bound algorithm starts on page 38, but the program employs none of the suggestions for improving computational efficiency. Only limited computational experience has been gained with this program. Very little can be said regarding the efficiency of the algorithm since only four problems have been solved with the algorithm. The problem sizes (job x machine), the number of iterations to arrive at a solution, the core storage requirements (in thousands of bytes), and the computing times (in seconds) are shown below. The first problem solved is the example problem discussed previously.

PROBLEM NUMBER	SIZE	NO. OF ITERATIONS	STORAGE	COMPUTING TIME
1	3x4	25	82	8.40
2	3x4	44	86	8.65
3	4x5	283	120	33.75
4	4x10	251	350	23.52

The process times and technological orderings for problems 3 and 4 were generated randomly. Problem 2 is taken from Brooks and White [Ref. 6].

V. CONCLUSION

Two methods for solving the n-job m-machine scheduling problem have been suggested. The graphical method for computing an upper bound on the time to process n jobs on m machines is simple to perform but does not always yield the optimal solution. The method does, however, yield an upper and lower bound on this schedule time. The optimal solution has been determined, of course, when these two bounds are the same. The branch and bound procedure described, on the other hand, always produces the optimal schedule if storage capacity and computation time permit. Its computational efficiency, however, must be examined in greater detail before it can be considered an efficient algorithm.

```
//SHA11033  JOB (1033,0562FT,ROL9),'SHACKELTON'
//     EXEC  FORTCLG,REGION.GO=102K
//FORT.SYSIN  DD *
C*****************************************************************
C
C            N-JOB M-MACHINE SCHEDULING PROBLEM
C
C               - BRANCH AND BOUND METHOD -
C
C         THIS PROGRAM UTILIZES THE BRANCH AND BOUND METHOD
C     TO SCHEDULE N JOBS ON M MACHINES SO THAT THE TIME TO
C     PROCESS ALL JOBS ON ALL MACHINES IS A MINIMUM.
C
C         THE JOBS ARE ORDERED 1,...,N IN ORDER OF THEIR
C     PRIORITIES.  THE MACHINE PROCESSING TIMES (TIME(JOB,
C     MACHINE)) AND TECHNOLOGICAL ORDERING OF MACHINES FOR
C     EACH JOB (IORDER(JOB,MACHINE)) MUST BE PLACED ON DATA
C     CARDS BY THE USER.  THE FORMATS ARE: F10.1 AND I10,
C     RESPECTIVELY.  THE NUMBER OF JOBS (N), THE NUMBER OF
C     MACHINES (M), AND THE TOTAL NUMBER OF ITERATIONS EX-
C     PECTED (ITERTO) MUST BE CHANGED FOR EACH DIFFERENT
C     PROBLEM.  IN ADDITION TO THESE CHANGES, THE STORAGE
C     ALLOTMENTS IN COMMON MUST BE CHANGED TO AGREE WITH THE
C     NEW N, M, AND ITERTO.
C
C         THE ALGORITHM OF SECTION IV.,B. OF THE THESIS IS
C     USED TO CALCULATE THE LOWER BOUNDS AND DEFINE THE
C     BRANCHING OPERATIONS.  AT EACH ITERATION THE BRANCHES
C     CONTINUE FROM THE LAST BRANCH WHICH HAD THE SMALLEST
C     LOWER BOUND.
C
C         THE SUBROUTINES CALLED ARE:
C
C         (1)  TABLO
C
C         TABLO INITIALIZES THE VECTORS START, TIME, AND
C     FINISH.  THE INITIAL TABLEAU IS THEREBY SPECIFIED.
C
C         (2)  XMAX
C
C         XMAX COMPUTES THE MAXIMUM OF ALL THE T(I)'S ASSOC-
C     IATED WITH THE PRESENT ITERATION.  THIS MAX(T(I)) IS
C     THE LOWER BOUND FOR THAT ITERATION.
C
C         (3)  PERMUT
C
C         PERMUT ARRANGES THE ORDER OF JOBS ON A MACHINE.
C     THIS ORDERING IS CHANGED WITH EACH BRANCHING OPERATION
C
C         (4)  CHANGE
C
C         CHANGE UPDATES THE TABLEAU OF A PARTICULAR ITER-
C     ATION SO THAT THE CONSTRAINTS ARE SATISFIED.  THE
C     EQUATIONS USED ARE THOSE SPECIFIED IN THE ALGORITHM.
C
C         (5)  BMIN
C
C         BMIN ORDERS THE LOWER BOUNDS OF ALL BRANCHES
C     THROUGH THE PRESENT ITERATION.  THE MINIMUM LOWER
C     BOUND (BDMIN) AND ITS ITERATION NUMBER (MIN) ARE RE-
C     TURNED TO THE MAIN PROGRAM.
C
C         (6)  OUTPUT
C
C         OUTPUT PRINTS OUT THE NUMBER OF JOBS PRESENTLY BE-
C     ING CONSIDERED, THE MACHINE ON WHICH THESE JOBS ARE
C     BEING PLACED, THE POSITION OF THE MOST RECENT JOB IN
C     THE ORDERING, AND THE LOWER BOUND ASSOCIATED WITH THAT
C     ORDERING.  THIS INFORMATION IS RECORDED FOR EACH
C     ITERATION.
C
C*****************************************************************
```

```
C     CHANGE THE VALUES OF ALLOCATION FOR EACH OF THE
C     VARIABLES IN COMMON TO AGREE WITH THE VALUES OF N, M,
C     AND ITERTO FOR THE PROBLEM BEING CONSIDERED.  THERE
C     ARE SEVEN IDENTICAL COMMON STATEMENTS; ONE IN THE MAIN
C     PROGRAM AND ONE IN EACH SUBROUTINE.  THEY MUST ALL
C     AGREE.  CHANGE ALSO THE ALLOCATION FOR JOB, MACH, AND
C     JUMP IN THE DIMENSION STATEMENT IN THE MAIN PROGRAM.
C
      COMMON START(3,4,30),TIME(3,4),T(3,30),FINISH(3,4,50),
     1BOUND(30),IORDER(3,4),JORDER(3,4)
      DIMENSION JOB(30),MACH(30),JUMP(3,4,30)
C
C     INSERT THE REQUIRED VALUES OF NUMBER OF JOBS, NUMBER
C     OF MACHINES, AND THE EXPECTED TOTAL NUMBER OF ITER-
C     ATIONS:
C
      N=3
      M=4
      ITERTO=30
C
C     INITIALIZE VARIABLES:
C
      ITRATE=1
      MIN=1
      DINFIN=999999.9
      INFIN=999999
      DO 137 INDEX=1,ITERTO
  137 JOB(INDEX)=2
      MACH(MIN)=1
C
C     SET ALL LOWER BOUNDS EQUAL TO INFINITY:
C
      DO 100 INDEX=1,ITERTO
  100 BOUND(INDEX)=DINFIN
C
C     READ THE INPUT DATA CARDS:
C
      DO 101 INDEX=1,N
      DO 101 JINDEX=1,M
  101 READ(5,102)TIME(INDEX,JINDEX),IORDER(INDEX,JINDEX)
  102 FORMAT(F10.1,I10)
      DO 141 INDEX=1,N
      DO 141 JINDEX=1,M
  141 JUMP(INDEX,JINDEX,1)=1
C
C     INITIALIZE THE FIRST TABLEAU:
C
      CALL TABLO(N,M)
C
C     FIND THE FIRST LOWER BOUND:
      CALL XMAX(1,N)
C
C     PRINT OUT THE TITLES IN THE OUTPUT:
C
      WRITE(6,117)
  117 FORMAT('1',T41,'BRANCH AND BOUND SOLUTION')
      WRITE(6,118)
  118 FORMAT('0',T11,'ITERATION',T31,'JOB LEVEL',T49,
     1'MACHINE LEVEL'T69,'POSITION OF I',T90,
     2'LOWER BOUND',//)
C
C     START THE ALGORITHM:
C
  111 I=JOB(MIN)
      II=I
  110 K=MACH(MIN)
      JO=1
  150 ITEM=I-1
      MACHIN=IORDER(I,K)
      DO 139 INDEX=1,ITEM
  139 JORDER(INDEX,K)=JUMP(INDEX,MACHIN,MIN)
C
```

```
C     GENERATE THE ORDERING OF JOBS ON MACHINE K:
C
      CALL PERMUT(I,K,JO)
      DO 135 ITEM=1,N
      DO 136 JTEM=1,M
      START(ITEM,JTEM,ITRATE+1)=START(ITEM,JTEM,MIN)
  136 FINISH(ITEM,JTEM,ITRATE+1)=FINISH(ITEM,JTEM,MIN)
  135 T(ITEM,ITRATE+1)=T( ITEM,MIN)
      DO 140 ITEM=1,N
      DO 140 JTEM=1,M
  140 JUMP(ITEM,JTEM,ITRATE+1)=JUMP(ITEM,JTEM,MIN)
      MACHIN=IORDER(I,K)
      DO 138 INDEX=1,I
  138 JUMP(INDEX,MACHIN,ITRATE+1)=JORDER(INDEX,K)
      JJ=1
  106 DO 105 III=1,I
      IF(JORDER(III,K).EQ.JJ)JJJ=III
      IF(JORDER(III,K).EQ.(JJ+1))KKK=III
  105 CONTINUE
C
C     CHECK THE JOB ORDERINGS AND CHANGE START AND FINISH
C     TIMES AS NECESSARY:
C
      DO 104 L=1,M
      IF(IORDER(II,K).EQ.IORDER(JJJ,L))J=L
      IF(IORDER(II,K).EQ.IORDER(KKK,L))IJ=L
  104 CONTINUE
      IF(FINISH(JJJ,J,ITRATE+1).GT.START(KKK,IJ,ITRATE+1))
     1GO TO 107
      MACH(ITRATE+1)=MACH(MIN)+1
      JOB(ITRATE+1)=JOB(MIN)
      JJ=JJ+1
      IF(JJ.LE.(I-1))GO TO 106
      GO TO 108
  107 CALL CHANGE(JJJ,KKK,J,IJ,ITRATE,M)
      MACH(ITRATE+1)=MACH(MIN)+1
      JOB(ITRATE+1)=JOB(MIN)
      JJ=JJ+1
      IF(JJ.LE.(I-1))GO TO 106
C
C     FIND MAX(T(I), I=1,...,N) FOR THIS ITERATION:
C
  108 ITER = ITRATE + 1
      CALL XMAX(ITER,N)
C
C     PRINT OUT THE INFORMATION FOR THIS ITERATION:
C
      CALL OUTPUT(ITER,I,M,JO,K)
C
C     START THE NEXT ITERATION:
C
      NN=I-1
      DO 109 III=1,NN
      IF(JORDER(III,K).GE.JO)JORDER(III,K)=JORDER(III,K)-1
  109 CONTINUE
      JO=JO+1
      ITRATE=ITRATE+1
      IF(JO.LE.I)GO TO 150
      ITER=ITRATE+1
      BOUND(1)=DINFIN
      CALL BMIN(ITER,MIN,BDMIN)
      BOUND(MIN)=DINFIN
      WRITE(6,112)MIN
  112 FORMAT(' ',T41,'BRANCH FROM ITERATION NUMBER ',I4,'.')
C
C     GO TO THE NEXT MACHINE FOR JOB I:
C
      IF(MACH(MIN).LE.M)GO TO 110
C
C     ATTEMPT TO SCHEDULE THE NEXT JOB:
C
      JOB(MIN)=JOB(MIN)+1
```

```
      MACH(MIN)=1
      IF(JOB(MIN).LE.N)GO TO 111
C
C     WRITE OUT THE OPTIMAL SOLUTION:
C
      WRITE(6,113)BDMIN,ITRATE
  113 FORMAT('1',T21,'THE OPTIMAL SOLUTION IS:',//,T31,
     1'MIN(MAX(T(I), I=1,...,N)) = ',F10.1,//,T21,
     2'THE SOLUTION REQUIRED ',I4,' ITERATIONS.')
      WRITE(6,114)
  114 FORMAT('0',T21,'THE FINAL TABLEAU IS:'//)
      DO 115 I=1,N
      DO 115 J=1,M
  115 WRITE(6,116)I,J,START(I,J,MIN),I,J,FINISH(I,J,MIN),
     1IORDER(I,J)
  116 FORMAT(' ',T11,'S(',I2,',',I2,')=',F10.1,T41,'F(',I2,
     1',',I2,')=',F10.1,T71,'MACHINE NUMBER ',I2)
      STOP
      END
C
C


      SUBROUTINE TABLO(N,M)
C
      COMMON START(3,4,30),TIME(3,4),T(3,30),FINISH(3,4,50),
     1BOUND(30),IORDER(3,4),JORDER(3,4)
C
      DO 201 I=1,N
      START(I,1,1)=0.0
      DO 200 J=2,M
      START(I,J,1)=START(I,J-1,1)+TIME(I,J-1)
      FINISH(I,J-1,1)=START(I,J,1)
  200 CONTINUE
      FINISH(I,M,1)=START(I,M,1)+TIME(I,M)
      T(I,1)=FINISH(I,M,1)
  201 CONTINUE
      RETURN
      END
C
C


      SUBROUTINE XMAX(ITRATE,N)
C
      COMMON START(3,4,30),TIME(3,4),T(3,30),FINISH(3,4,50),
     1BOUND(30),IORDER(3,4),JORDER(3,4)
C
      BOUND1=0.0
      DO 300 I=1,N
      IF(BOUND1.GT.T(I,ITRATE))GO TO 300
      BOUND1=T(I,ITRATE)
  300 CONTINUE
      BOUND(ITRATE)=BOUND1
      RETURN
      END
C
C


      SUBROUTINE PERMUT(I,K,JO)
C
      COMMON START(3,4,30),TIME(3,4),T(3,30),FINISH(3,4,50),
     1BOUND(30),IORDER(3,4),JORDER(3,4)
C
      JORDER(I,K)=JO
      NN=I-1
      DO 400 J=1,NN
      IF(JORDER(J,K).GE.JO)JORDER(J,K)=JORDER(J,K)+1
  400 CONTINUE
      RETURN
      END
```

```
C
C
      SUBROUTINE CHANGE(I,J,K,L,ITRATE,M)
C
      COMMON START(3,4,30),TIME(3,4),T(3,30),FINISH(3,4,50),
     1BOUND(30),IORDER(3,4),JORDER(3,4)
C
      DELTA=FINISH(I,K,ITRATE+1)-START(J,L,ITRATE+1)
      NORM=START(J,L,ITRATE+1)
      DO 500 II=1,M
      IF(START(J,II,ITRATE+1).LT.NORM)GO TO 500
      START(J,II,ITRATE+1)=START(J,II,ITRATE+1)+DELTA
      FINISH(J,II,ITRATE+1)=START(J,II,ITRATE+1)+TIME(J,II)
  500 CONTINUE
      T(J,ITRATE+1)=FINISH(J,M,ITRATE+1)
      RETURN
      END
C
C
      SUBROUTINE BMIN(ITERTO,MIN,BDMIN)
C
      COMMON START(3,4,30),TIME(3,4),T(3,30),FINISH(3,4,50),
     1BOUND(30),IORDER(3,4),JORDER(3,4)
C
      DIMENSION W(30)
      DO 602 II=1,ITERTO
  602 W(II)=BOUND(II)
      NPASS=ITERTO-1
      DO 600 I=1,NPASS
      NSTOP=ITERTO-I
      DO 600 J=1,NSTOP
      IF(W(J).LE.W(J+1))GO TO 600
      TEMP=W(J)
      W(J)=W(J+1)
      W(J+1)=TEMP
  600 CONTINUE
      DO 603 II=1,ITERTO
      IF(W(1).EQ.BOUND(II))MIN=II
  603 CONTINUE
      BDMIN=BOUND(MIN)
      RETURN
      END
C
C
      SUBROUTINE OUTPUT(ITRATE,I,M,JO,K)
C
      COMMON START(3,4,30),TIME(3,4),T(3,30),FINISH(3,4,50),
     1BOUND(30),IORDER(3,4),JORDER(3,4)
C
      WRITE(6,700)ITRATE,I,IORDER(I,K),JO,BOUND(ITRATE)
  700 FORMAT(' ',T12,I4,T35,I2,T54,I2,T74,I2,T93,F10.2)
      RETURN
      END
//GO.SYSIN  DD *
```

ITERATION	JOB LEVEL	MACHINE LEVEL	POSITION OF I	LOWER BOUND
2	2	2	1	21.00
3	2	2	2	21.00
	BRANCH FROM ITERATION NUMBER 3.			
4	2	1	1	36.00
5	2	1	2	21.00
	BRANCH FROM ITERATION NUMBER 5.			
6	2	4	1	25.00
7	2	4	2	23.00
	BRANCH FROM ITERATION NUMBER 2.			
8	2	1	1	32.00
9	2	1	2	21.00
	BRANCH FROM ITERATION NUMBER 9.			
10	2	4	1	21.00
11	2	4	2	25.00
	BRANCH FROM ITERATION NUMBER 10.			
12	2	3	1	28.00
13	2	3	2	21.00
	BRANCH FROM ITERATION NUMBER 13.			
14	3	1	1	25.00
15	3	1	2	21.00
16	3	1	3	24.00
	BRANCH FROM ITERATION NUMBER 15.			
17	3	3	1	22.00
18	3	3	2	24.00
19	3	3	3	28.00
	BRANCH FROM ITERATION NUMBER 17.			
20	3	4	1	22.00
21	3	4	2	24.00
22	3	4	3	29.00
	BRANCH FROM ITERATION NUMBER 20.			
23	3	2	1	38.00
24	3	2	2	33.00
25	3	2	3	22.00

THE OPTIMAL SOLUTION IS:

MIN(MAX(T(I), I=1,...,N)) = 22.00

THE SOLUTION REQUIRED 25 ITERATIONS.

THE FINAL TABLEAU IS:

S(1,1)= 0.0	F(1,1)= 3.0	MACHINE NUMBER 1
S(1,2)= 5.0	F(1,2)= 8.0	MACHINE NUMBER 2
S(1,3)= 9.0	F(1,3)= 16.0	MACHINE NUMBER 3
S(1,4)= 16.0	F(1,4)= 22.0	MACHINE NUMBER 4
S(2,1)= 0.0	F(2,1)= 5.0	MACHINE NUMBER 2
S(2,2)= 7.0	F(2,2)= 13.0	MACHINE NUMBER 1
S(2,3)= 13.0	F(2,3)= 15.0	MACHINE NUMBER 4
S(2,4)= 16.0	F(2,4)= 18.0	MACHINE NUMBER 3
S(3,1)= 3.0	F(3,1)= 7.0	MACHINE NUMBER 1
S(3,2)= 7.0	F(3,2)= 9.0	MACHINE NUMBER 3
S(3,3)= 9.0	F(3,3)= 12.0	MACHINE NUMBER 4
S(3,4)= 12.0	F(3,4)= 16.0	MACHINE NUMBER 2

BIBLIOGRAPHY

1. Hardgrave, W. W. and Nemhauser, G. L., "A Geometric Model and a Graphical Algorithm for a Sequencing Problem," Operations Research, v. 11, No. 6, p. 889-900, November 1963.

2. Sisson, R. L., "Sequencing Theory," Progress in Operations Research, v. 1, Ackoff, R. L., ed., Chapter 7, Hohn Wiley, 1961.

3. Conway, R. W., Maxwell, W. L., and Miller, L. W., Theory of Scheduling, Addison-Wesley, 1967.

4. Akers, S. B., Jr., and Friedman, J., "A Non-Numerical Approach to Production Scheduling Problems," Operations Research, v. 3, No. 5, p. 429-442, November 1955.

5. Szwarc, W., "On Some Sequencing Problems," Naval Research Logistics Quarterly, v. 15, No. 2, p. 127-155, June 1968.

6. Brooks, G. H., and White, C. R., "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem," Journal of Industrial Engineering, v. 16, No. 1, p. 34-40, January 1965.

7. Ignall, E., and Schrage, L., "Application of the Branch and Bound Technique to Some Flowshop Scheduling Problems," Operations Research, v. 13, No. 3, p. 400-412, May 1965.

8. Little, J. D. C., and others, "An Algorithm for the Traveling Salesman Problem," Operations Research, v. 8, No. 2, p. 972-989, November 1963.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst Professor G. T. Howard, Code 55Hk Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
4.	LT Norman J. Shackelton Jr., USN SMC 1092 Naval Postgraduate School Monterey, California 93940	1
5.	Department of Operations Analysis (Code 55) Naval Postgraduate School Monterey, California 93940	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

Approaches to the n-Job m-Machine Scheduling Problem

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; March 1971

5. AUTHOR(S) (First name, middle initial, last name)

Norman John Shackelton, Jr.

6. REPORT DATE March 1971	7a. TOTAL NO. OF PAGES 47	7b. NO. OF REFS 8
8a. CONTRACT OR GRANT NO. b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

This thesis presents two methods of solving the n-job m-machine job shop scheduling problem. The criterion for optimality is the minimization of the total time to process all jobs on all machines. The technological ordering of machines for each job is fixed, known, and nonrandom.

The first method presented, a graphical method, indicates a lower bound and an upper bound on the optimal time to process all jobs on all machines.

The second method is a branch and bound algorithm. In principle an optimal solution can always be determined by this method. Only limited computational experience is presented for the algorithm but some methods for efficient computation are suggested.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
job shop scheduling						
branch and bound						
graphical solution						

DD FORM 1 NOV 65 1473 (BACK)

S/N 0101-807-6821